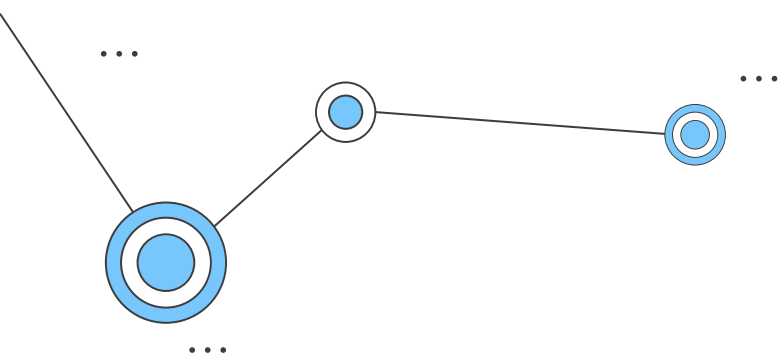


Git e Github

Módulo 1 - Introdução ao
controle de versão



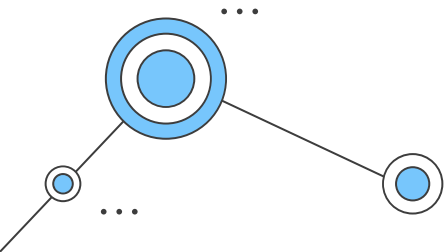
Objetivo do curso

Aprender como controlar as versões de seus códigos e arquivos de configuração usando sistemas de controle de versão, em especial o git.

Apresentar e demonstrar o funcionamento da plataforma de versionamento Github

Bônus:

- GitHub Student Developer Pack,
- "release the kraken"
- sorteios



01

Introdução ao controle de versão

v1 ... v2 ... v3 ... final ...

Pode não parecer, mas isso não é certo!

02

Usando o GIT localmente

Primeiros passos para saber onde foi parar aquele código que funcionava.

03

Introdução ao GITHUB

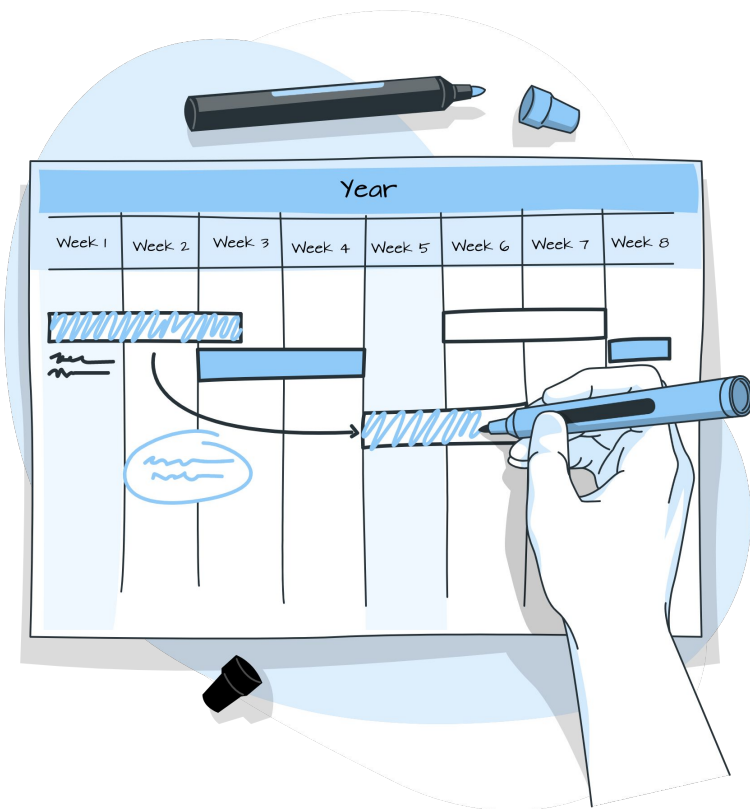
Sim, a Microsoft controla todos os
códigos abertos do mundo...

04

Colaboração

Se não dá pra fazer sozinho, é melhor saber como fazer junto...

Nossa agenda





Ambiente e ferramentas utilizadas



Não se preocupe. Tudo o que vamos usar é multi-plataforma e terão links e tutoriais no SIGA

Mas, afinal, porquê estou aqui?



- Desenvolvedor desde os 14 anos (já faz muito tempo...não precisa contar)
- Trabalhou para/por Precon, Usiminas, Vallourec & Mannesmann, UFMG, Gerdau, Ford e AmBev
- Sócio fundador da Virturian Ativos Industriais
- Supervisor de desenvolvimento no UNIFAGOC

01 Introdução ao controle de versão

01 – Introdução ao controle de versão

Aplicação

Artefatos de desenvolvimento e demais arquivos relativos à aplicação

Infraestrutura

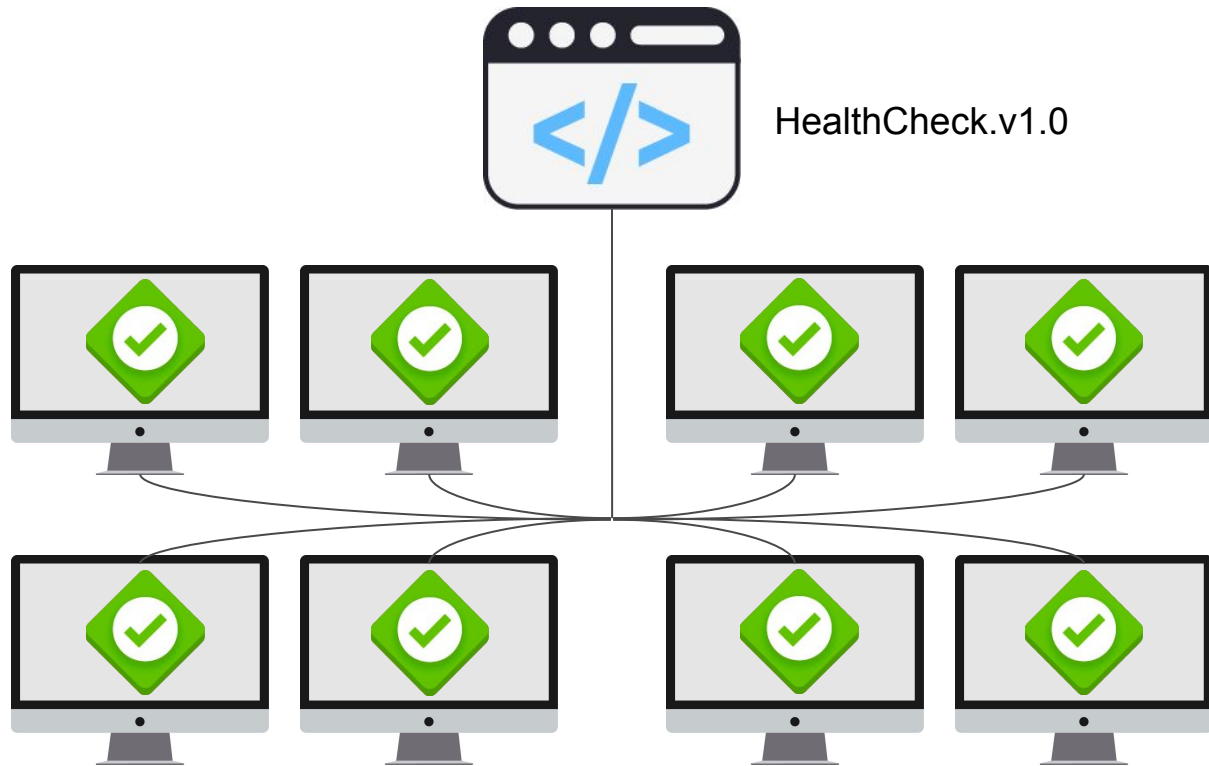
Configurações de máquinas, rotas de rede, ip's e MAC's dos equipamentos...

Documentação

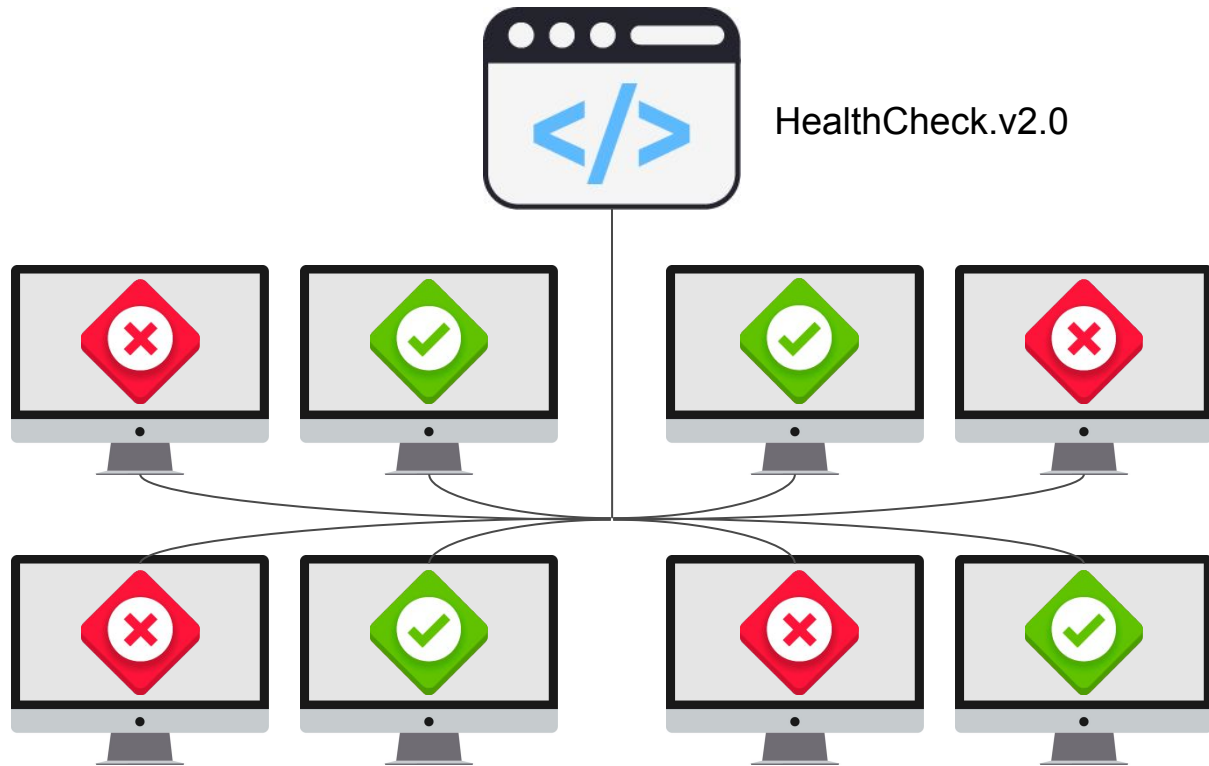
Histórico de modificações de requisitos de software e regras de negócio



01 – Introdução ao controle de versão



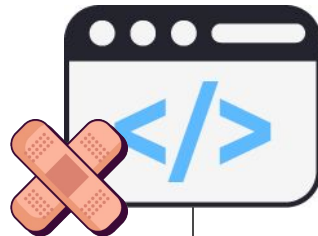
01 – Introdução ao controle de versão



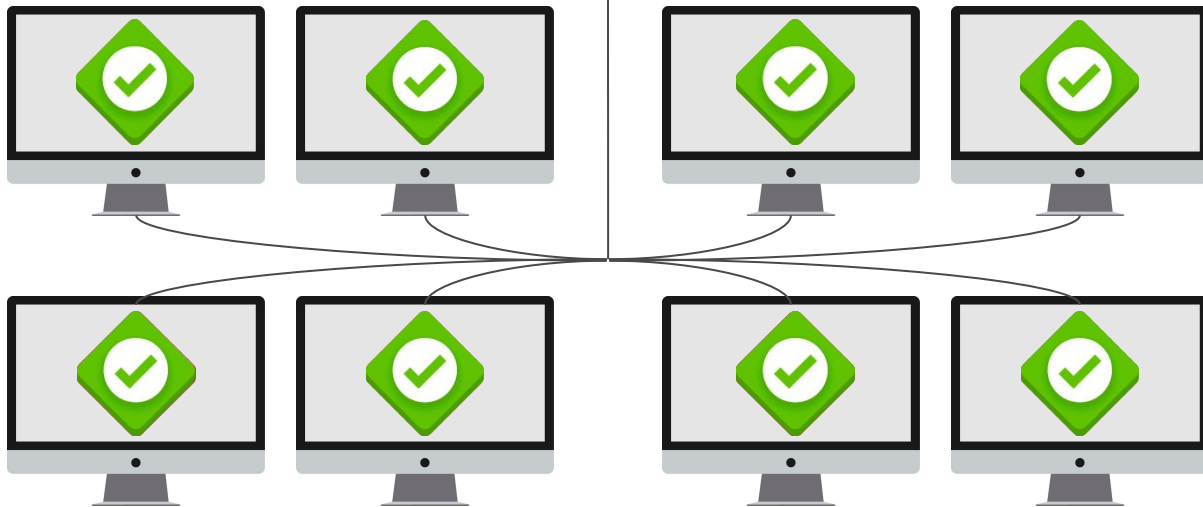
01 – Introdução ao controle de versão



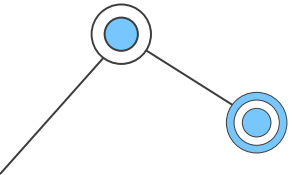
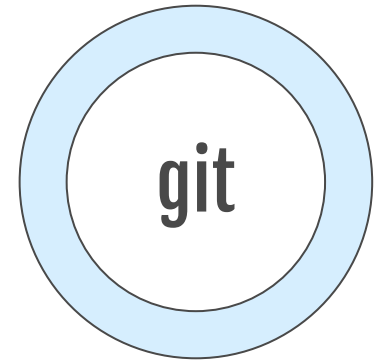
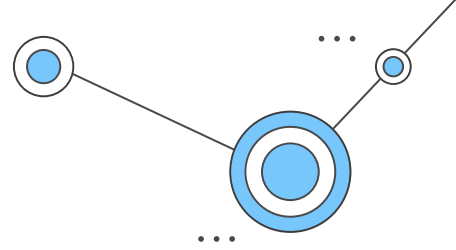
Sistema de
Controle de Versão



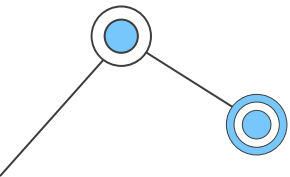
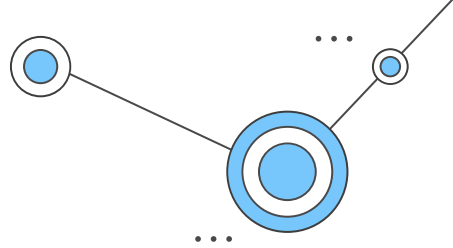
HealthCheck.v2.0



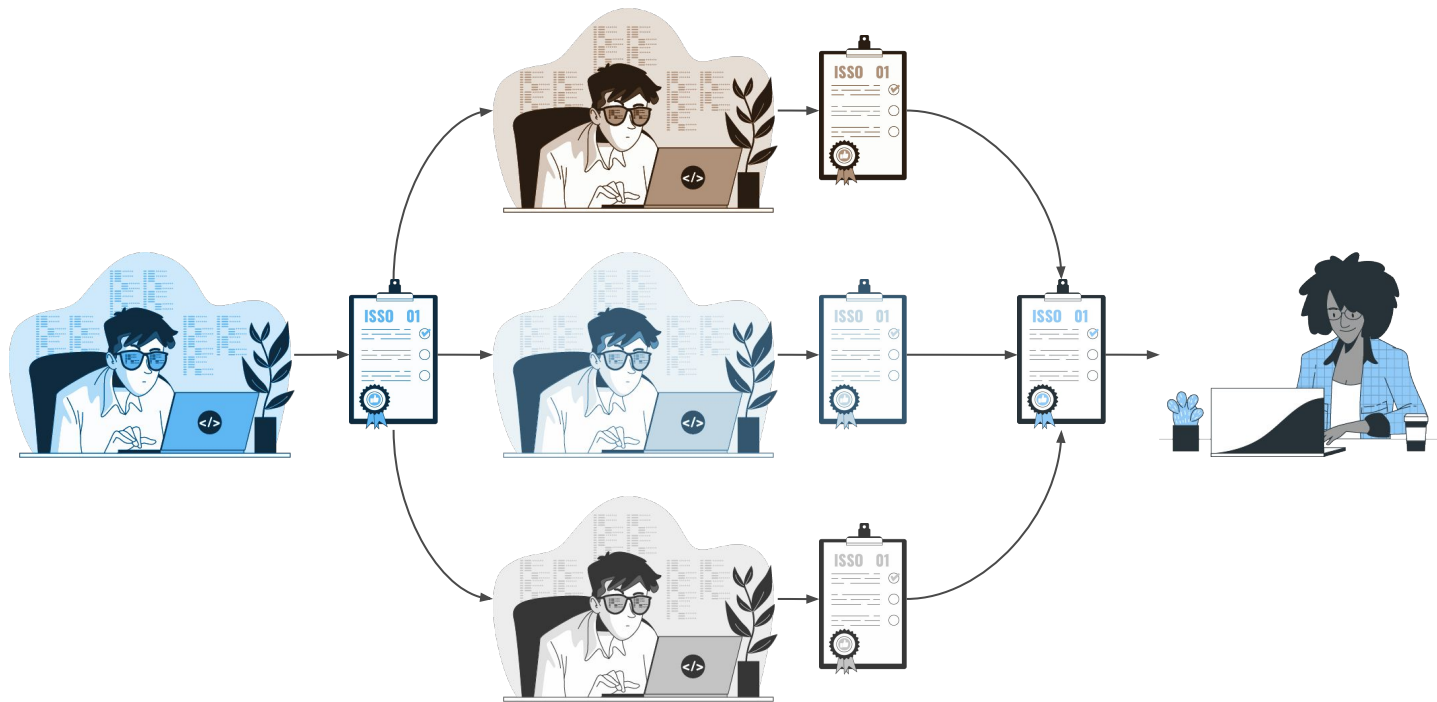
01 – Introdução ao controle de versão



01 - Introdução ao controle de versão



01 – Introdução ao controle de versão



Histórico de Cópias



01 – Introdução ao controle de versão

Histórico de Cópias



Manual

Todo o controle de o que foi feito tem que ser acompanhado por uma pessoa

...



Alto custo

Mesmo que você tenha modificado apenas uma pequena parte, você precisa manter todo o histórico

...



Pouco detalhado

Saber quem fez qual parte exige muito esforço, tempo e comunicação

...

01 – Introdução ao controle de versão

```
1.py
D: > carlo > Documents > Git e Github > Semana 1 > 1.py
1  #!/usr/bin/env python3
2
3  import re
4
5  def rearrange_name(name):
6      result = re.search(r"^([\w .]*), ([\w .]*)$", name)
7      if result == None:
8          return result
9      return "{} {}".format(result[2], result[1])
10
```

```
2.py
D: > carlo > Documents > Git e Github > Semana 1 > 2.py
1  #!/usr/bin/env python3
2
3  import re
4
5  def rearrange_name(name):
6      result = re.search(r"^([\w .-]*), ([\w .-]*)$", name)
7      if result == None:
8          return result
9      return "{} {}".format(result[2], result[1])
10
```


01 – Introdução ao controle de versão

```
carlos in ~/CURSOS/GIT E GITHUB  
> diff 1.py 2.py
```

```
carlos in ~/CURSOS/GIT E GITHUB
> cat 1.py
#!/usr/bin/env python3
```

```
import re
```

```
def rearrange_name(name):
    result = re.search(r"^([\w .]*), ([\w .]*)$", name)
    if result == None:
        return result
    return "{} {}".format(result[2], result[1])
```

```
%
carlos in ~/CURSOS/GIT E GITHUB
> cat 2.py
#!/usr/bin/env python3
```

```
import re
```

```
def rearrange_name(name):
    result = re.search(r"^([\w .-]*), ([\w .-]*)$", name)
    if result == None:
        return result
    return "{} {}".format(result[2], result[1])
```

```
%
carlos in ~/CURSOS/GIT E GITHUB
> diff 1.py 2.py
```

```
6c6
< result = re.search(r"^([\w .]*), ([\w .]*)$", name)
---
> result = re.search(r"^([\w .-]*), ([\w .-]*)$", name)
carlos in ~/CURSOS/GIT E GITHUB
> |
```

Agora ficou
fácil, né?

01 – Introdução ao controle de versão

lao

```
1 The Way that can be told of is not the eternal Way;  
2 The name that can be named is not the eternal name.  
3 The Nameless is the origin of Heaven and Earth;  
4 The Named is the mother of all things.  
5 Therefore let there always be non-being,  
6 | so we may see their subtlety,  
7 And let there always be being,  
8 | so we may see their outcome.  
9 The two are the same,  
10 But after they are produced,  
11 | they have different names.  
12 The door of all subtleties!
```

tzu

```
1 The Nameless is the origin of Heaven and Earth;  
2 The named is the mother of all things.  
3  
4 Therefore let there always be non-being,  
5 | so we may see their subtlety,  
6 And let there always be being,  
7 | so we may see their outcome.  
8 The two are the same,  
9 But after they are produced,  
10 | they have different names.  
11 They both may be called deep and profound.  
12 Deeper and more profound,  
13 The door of all subtleties!
```

01 – Introdução ao controle de versão

```
carlos in ~/CURSOS/GIT E GITHUB  
> diff lao tzu
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> diff lao tzu
```

```
1,2d0
```

```
< The Way that can be told of is not the eternal Way;
```

```
< The name that can be named is not the eternal name.
```

```
4c2,3
```

```
< The Named is the mother of all things.
```

```
---
```

```
> The named is the mother of all things.
```

```
>
```

```
11a11,12
```

```
> They both may be called deep and profound.
```

```
> Deeper and more profound,
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> |
```

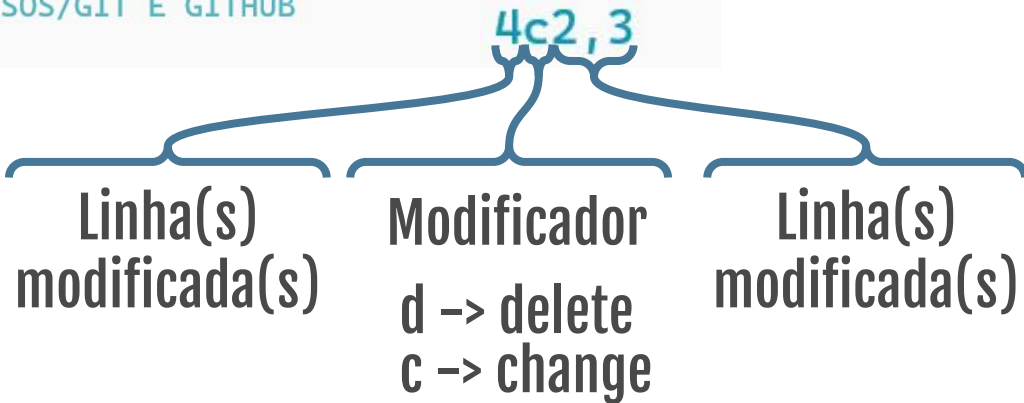
1,2d0

Linha(s)
modificada(s)

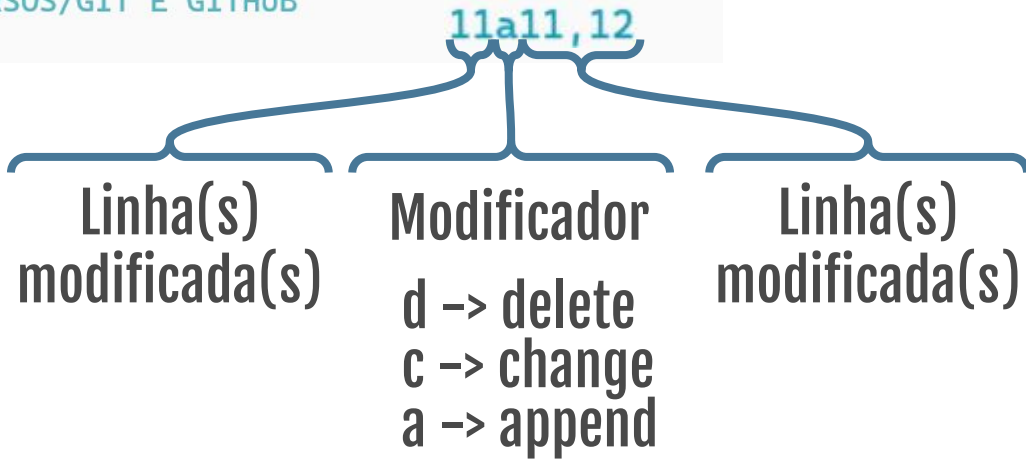
Modificador
d -> delete

Linha(s)
modificada(s)

```
carlos in ~/CURSOS/GIT E GITHUB
> diff lao tzu
1,2d0
< The Way that can be told of is not the eternal Way;
< The name that can be named is not the eternal name.
4c2,3
---
> The named is the mother of all things.
>
11a11,12
> They both may be called deep and profound.
> Deeper and more profound,
carlos in ~/CURSOS/GIT E GITHUB
> |
```



```
carlos in ~/CURSOS/GIT E GITHUB
> diff lao tzu
1,2d0
< The Way that can be told of is not the eternal Way;
< The name that can be named is not the eternal name.
4c2,3
< The Named is the mother of all things.
---
> The named is the mother of all things.
>
11a11,12
> They both may be called deep and profound.
> Deeper and more profound,
carlos in ~/CURSOS/GIT E GITHUB
> |
```



01 – Introdução ao controle de versão

```
carlos in ~/CURSOS/GIT E GITHUB  
> diff lao tzu -u
```




```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> diff lao tzu -u
```

```
--- lao 2021-10-29 09:04:32.313909500 -0300
```

```
+++ tzu 2021-10-29 09:03:38.703909500 -0300
```

```
@@ -1,7 +1,6 @@
```

```
-The Way that can be told of is not the eternal Way;
```

```
-The name that can be named is not the eternal name.
```

```
    The Nameless is the origin of Heaven and Earth;
```

```
-The Named is the mother of all things.
```

```
+The named is the mother of all things.
```

```
+
```

```
    Therefore let there always be non-being,
```

```
        so we may see their subtlety,
```

```
    And let there always be being,
```

```
@@ -9,4 +8,6 @@
```

```
    The two are the same,
```

```
    But after they are produced,
```

```
        they have different names.
```

```
+They both may be called deep and profound.
```

```
+Deeper and more profound,
```

```
    The door of all subtleties!
```

```
\ No newline at end of file
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> |
```

01 - Introdução ao controle de versão

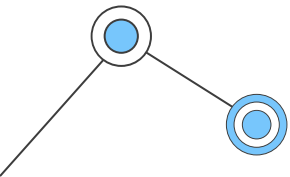
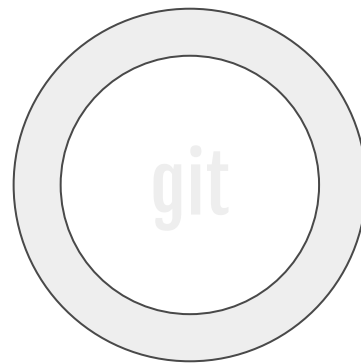
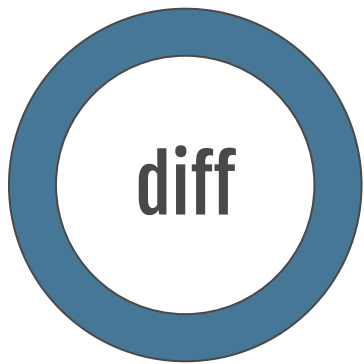
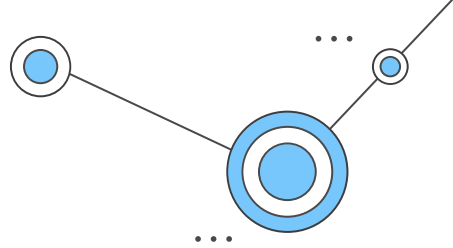
```
1 +-- 3 lines: #!/usr/bin/perl-----
4 use warnings;
5
6 #inserted a line here!
7
8 unchanged line
9
10 changed line
11
12 use DBI;
13
14
15 Added a line
16 -----
17 # Vars for db
18 #2345678901234567890
19     my $dbfile;
20         my $dsn;
21     my $user;
22 +-- 17 lines: my $password;-----
39 my $userID;
40 my $userName;
41 my $comment;
42
43
44
45 remove a linea

4 use warnings;
5
6 #inserted a line here!
7
8 unchanged line
9
10 changed this line - This line was changed
11
12 use DBI;
13
14
15 Added a line
16 Added a second line
17
18 # Vars for db
19 #2345678901234567890
20     my $dbfile;
21         my $dsn;
22     my $user;
23 +-- 17 lines: my $password;-----
40 my $userID;
41 my $userName;
42 my $comment;
43
44
45 remove a linea
```

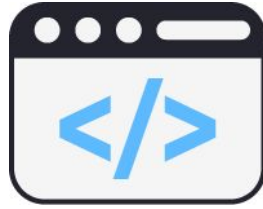
sqliteUpdate.pl 4,1 Top sqliteUpdate-updated.pl 4,1 2%

Number of remaining unsolved conflicts: 2 (of which 0 are whitespace)

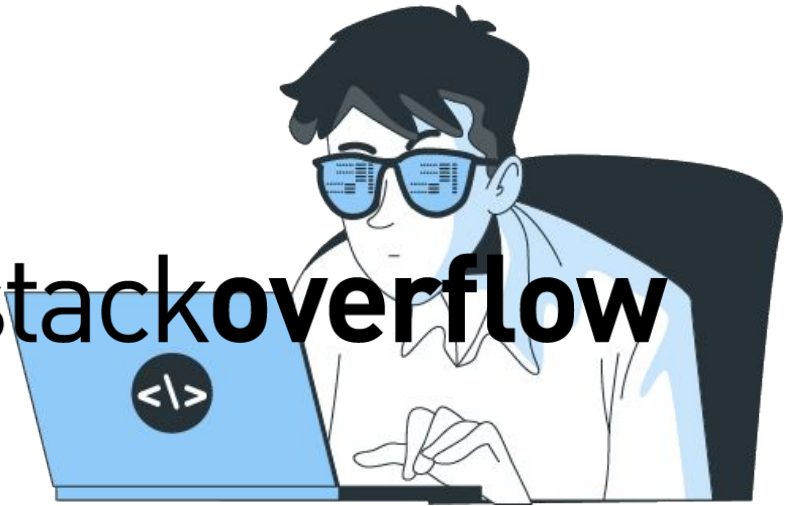
01 – Introdução ao controle de versão



01 – Introdução ao controle de versão



stackoverflow

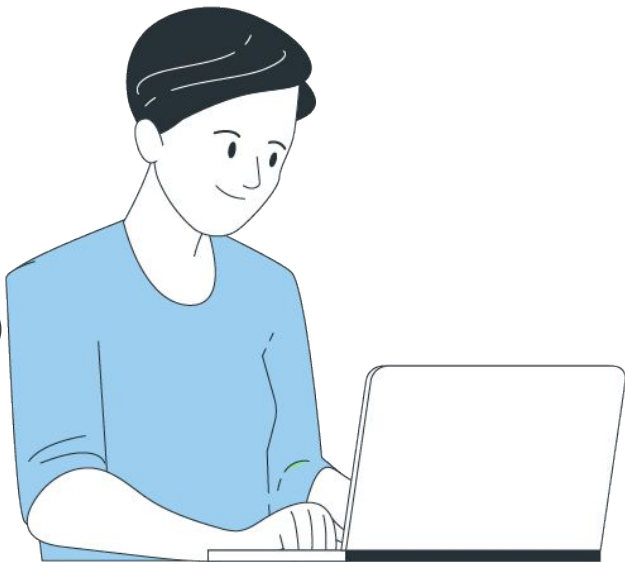


01 – Introdução ao controle de versão

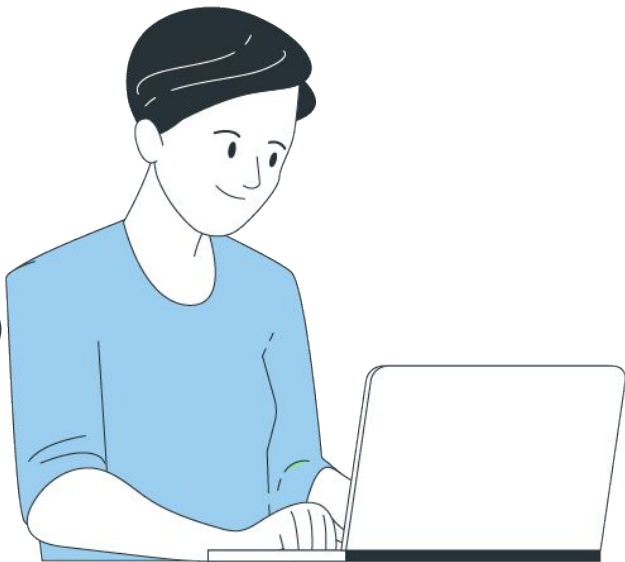


01 – Introdução ao controle de versão

```
> diff -u HealthCheck.v2.5.sh HealthCheck.v3.0.sh > mudancas.diff
```



01 – Introdução ao controle de versão



```
> patch cpu_usage.py < cpu_usage.diff
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> cat cpu_usage.py  
#!/usr/bin/env python3  
import psutil
```

```
def check_cpu_usage(percent):
```

```
    usage = psutil.cpu_percent()  
    return usage < percent
```

```
if not check_cpu_usage(75):  
    print("ERROR! CPU is overloaded")  
else:  
    print("Everything is awesome!")%
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> patch cpu_usage.py < cpu_usage.diff  
patching file cpu_usage.py
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> cat cpu_usage.py  
#!/usr/bin/env python3  
import psutil
```

```
def check_cpu_usage(percent):
```

```
    usage = psutil.cpu_percent(1)  
    print("DEBUG: usage: {}".format(usage))  
    return usage < percent
```

```
if not check_cpu_usage(75):  
    print("ERROR! CPU is overloaded")  
else:  
    print("Everything is awesome!")%
```

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> cat cpu_usage.diff  
--- cpu_usage.py          2021-10-29 10:29:06.243909500 -0300  
+++ cpu_usage_correct.py  2021-10-29 10:31:37.023909500 -0300  
@@ -2,7 +2,8 @@  
     import psutil
```

```
def check_cpu_usage(percent):
```

```
- usage = psutil.cpu_percent()  
+ usage = psutil.cpu_percent(1)  
+ print("DEBUG: usage: {}".format(usage))  
    return usage < percent
```

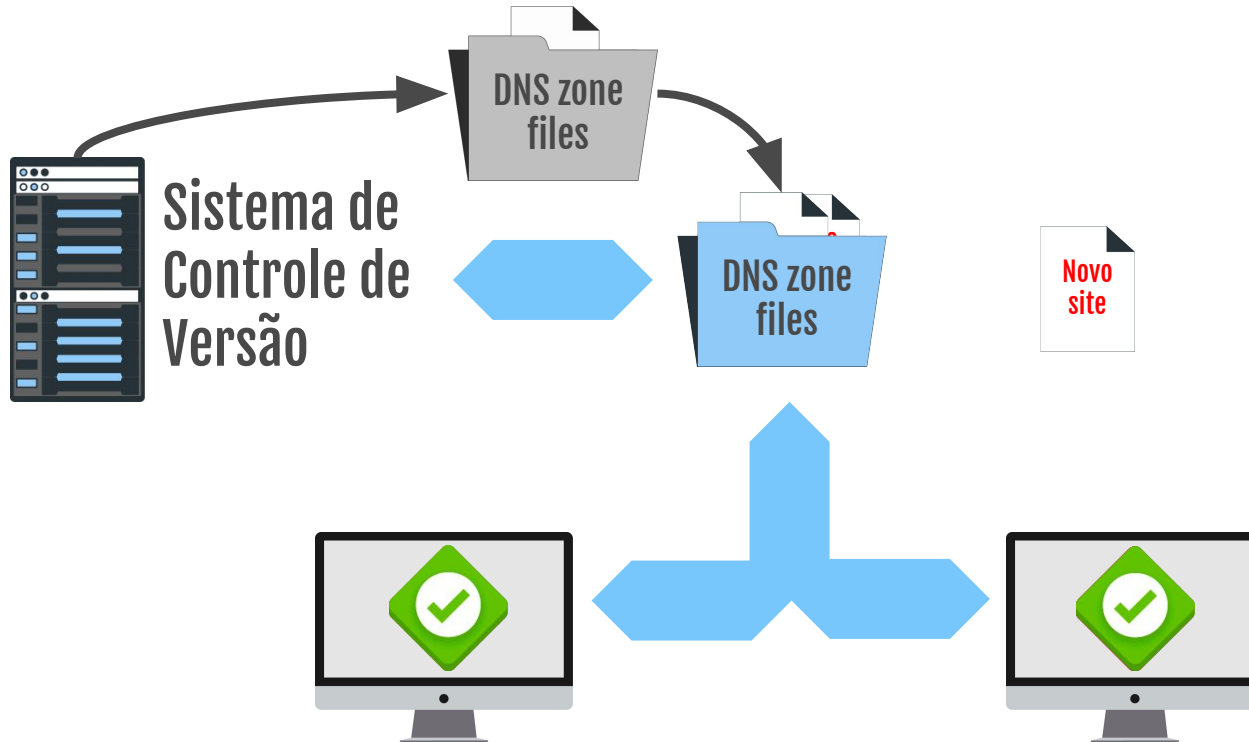
```
if not check_cpu_usage(75):
```

```
carlos in ~/CURSOS/GIT E GITHUB  
> |
```

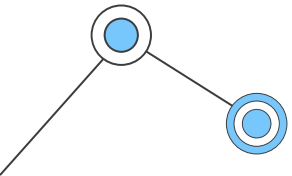
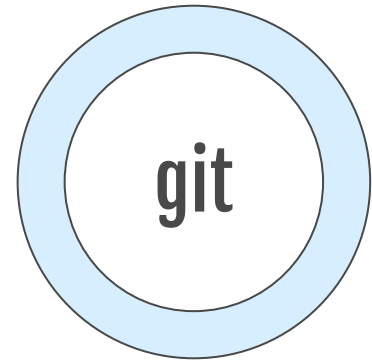
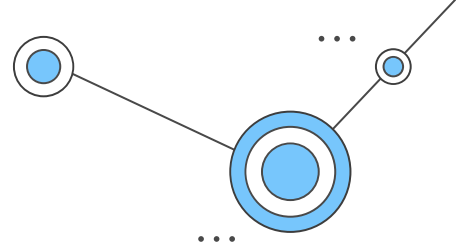

Sistema de Controle de Versão



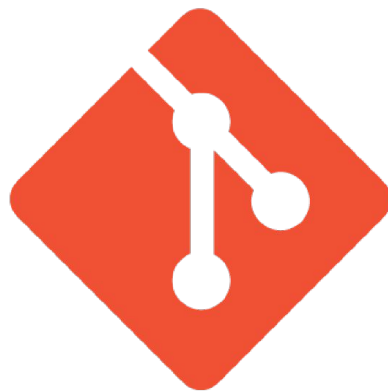
01 - Introdução ao controle de versão



01 – Introdução ao controle de versão

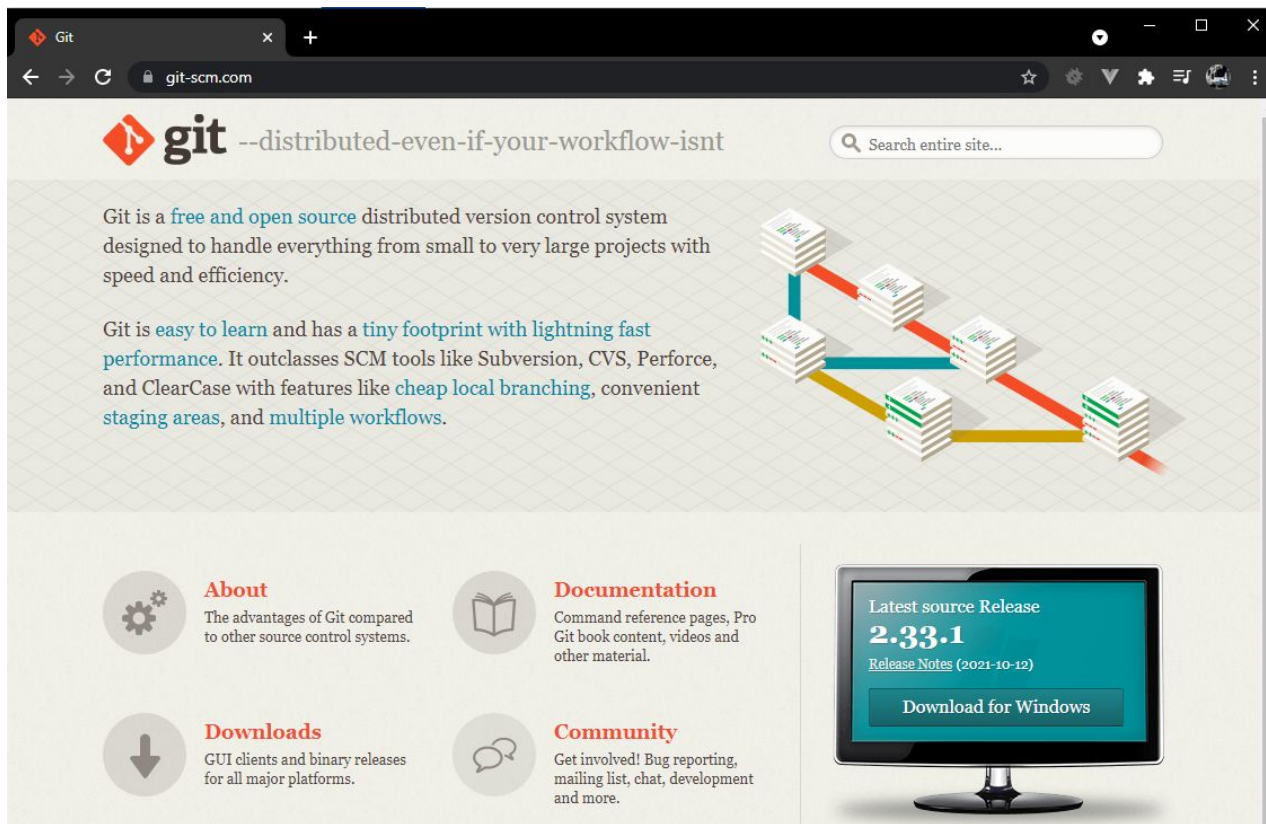


01 – Introdução ao controle de versão



git

01 – Introdução ao controle de versão



The image is a screenshot of the Git website homepage, viewed in a web browser. The browser's address bar shows 'git-scm.com'. The page features the Git logo and the tagline '--distributed-even-if-your-workflow-isnt'. A search bar is located in the top right. The main content area describes Git as a free and open source distributed version control system. To the right of the text is a diagram showing a sequence of code commits represented as stacks of papers, connected by colored lines (red, blue, yellow) indicating branching and merging. Below the main text are four sections: 'About' (advantages), 'Documentation' (reference pages), 'Downloads' (GUI clients), and 'Community' (bug reporting). In the bottom right corner, there is a monitor displaying the latest source release '2.33.1' and a 'Download for Windows' button.

Git --distributed-even-if-your-workflow-isnt

Git is a [free and open source](#) distributed version control system designed to handle everything from small to very large projects with speed and efficiency.

Git is [easy to learn](#) and has a [tiny footprint with lightning fast performance](#). It outclasses SCM tools like Subversion, CVS, Perforce, and ClearCase with features like [cheap local branching](#), convenient [staging areas](#), and [multiple workflows](#).

About
The advantages of Git compared to other source control systems.

Documentation
Command reference pages, Pro Git book content, videos and other material.

Downloads
GUI clients and binary releases for all major platforms.

Community
Get involved! Bug reporting, mailing list, chat, development and more.

Latest source Release
2.33.1
[Release Notes \(2021-10-12\)](#)
[Download for Windows](#)

01 – Introdução ao controle de versão



mercurial

Instalando o GIT



01 – Introdução ao controle de versão

```
carlos in ~  
> git --version  
git version 2.25.1
```

```
PS C:\Users\carlo> git --version  
git version 2.33.0.windows.2  
PS C:\Users\carlo> |
```


01 – Introdução ao controle de versão

```
carlos in ~  
> git config --global user.email "carlos.junior@unifagoc.edu.br"  
carlos in ~  
> git config --global user.name "Carlos Roberto Barreto Junior"
```

01 – Introdução ao controle de versão



Repositório
Local

The diagram consists of two large, light-blue circles with thick borders, positioned side-by-side. Each circle contains text. The left circle is labeled 'Repositório Local' and the right circle is labeled 'Repositório Remoto'. The circles are set against a light gray background with decorative vertical lines and small circular icons on the left and right sides.

Repositório
Remoto

01 – Introdução ao controle de versão

```
carlos in ~/CURSOS/GIT E GITHUB
```

```
> git init
```

```
Initialized empty Git repository in /home/carlos/CURSOS/GIT E GITHUB/.git/
```

Diretório GIT



Árvore de Trabalho



01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB  
> git add cpu_usage.py
```

01 – Introdução ao controle de versão

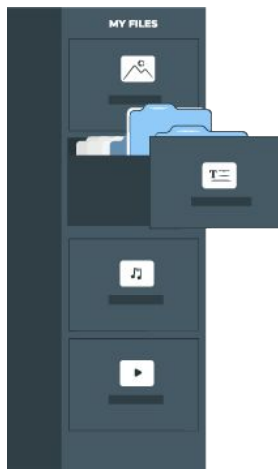
Staging Area (Index)

É uma estrutura de arquivos mantida pelo GIT que contém todas as informações sobre quais arquivos e mudanças serão enviados no próximo commit

01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB  
> git add cpu_usage.py
```

Diretório
GIT



Staging
Area



Árvore de
Trabalho



01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB
```

```
> git status
```

```
On branch master
```

```
No commits yet
```

```
Changes to be committed:
```

```
(use "git rm --cached <file>..." to unstage)
```

```
new file:   cpu_usage.py
```



01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB  
> git commit|
```



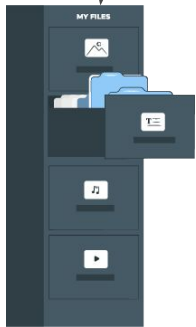
01 – Introdução ao controle de versão

untracked

committed

staged

modified



01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB on □ master
> ls -l
total 4
-rw-r--r-- 1 carlos carlos 228 Oct 30 10:05 cpu_usage.py
carlos in GIT E GITHUB on □ master
> git status
On branch master
nothing to commit, working tree clean
```

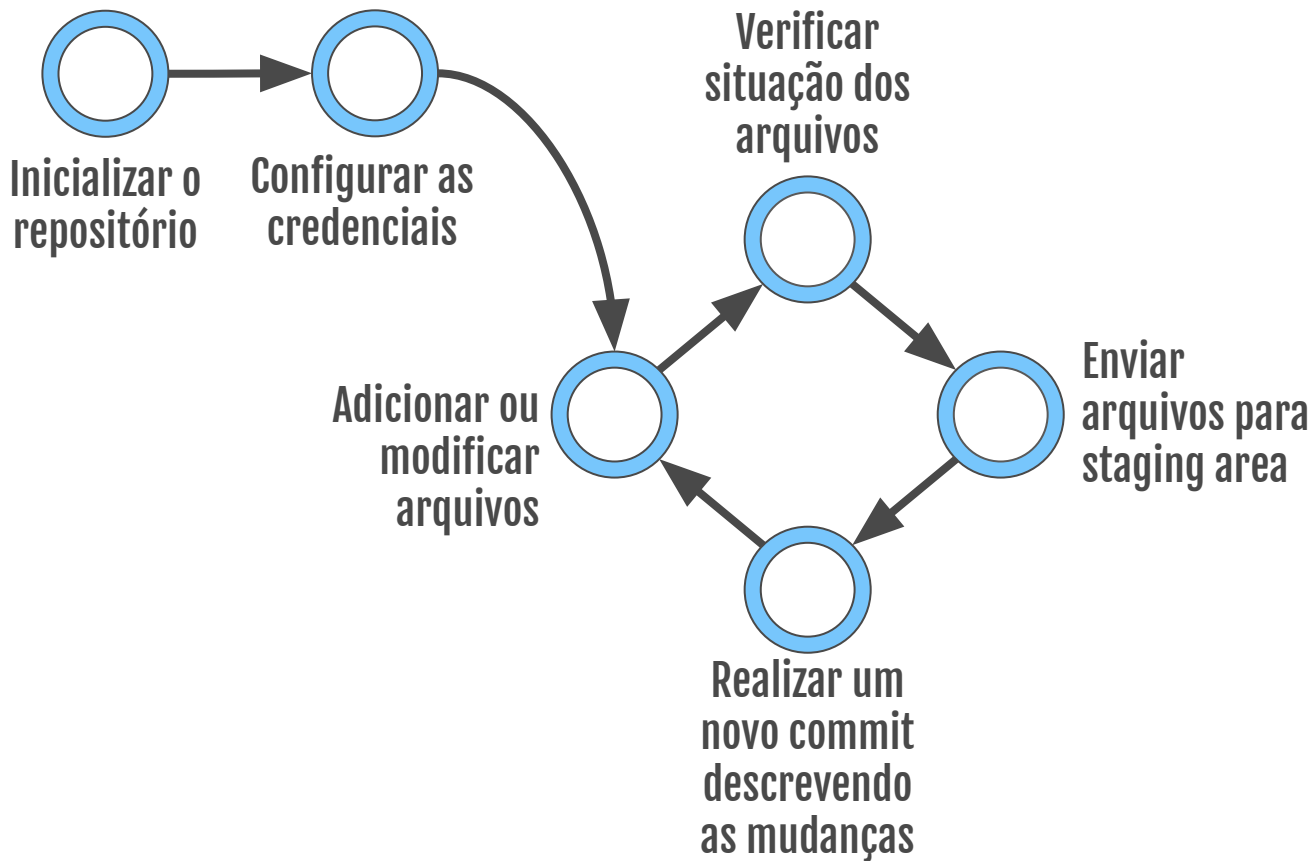
01 – Introdução ao controle de versão

```
carlos in GIT E GITHUB on ◻ master [!]  
> git add cpu_usage.py
```

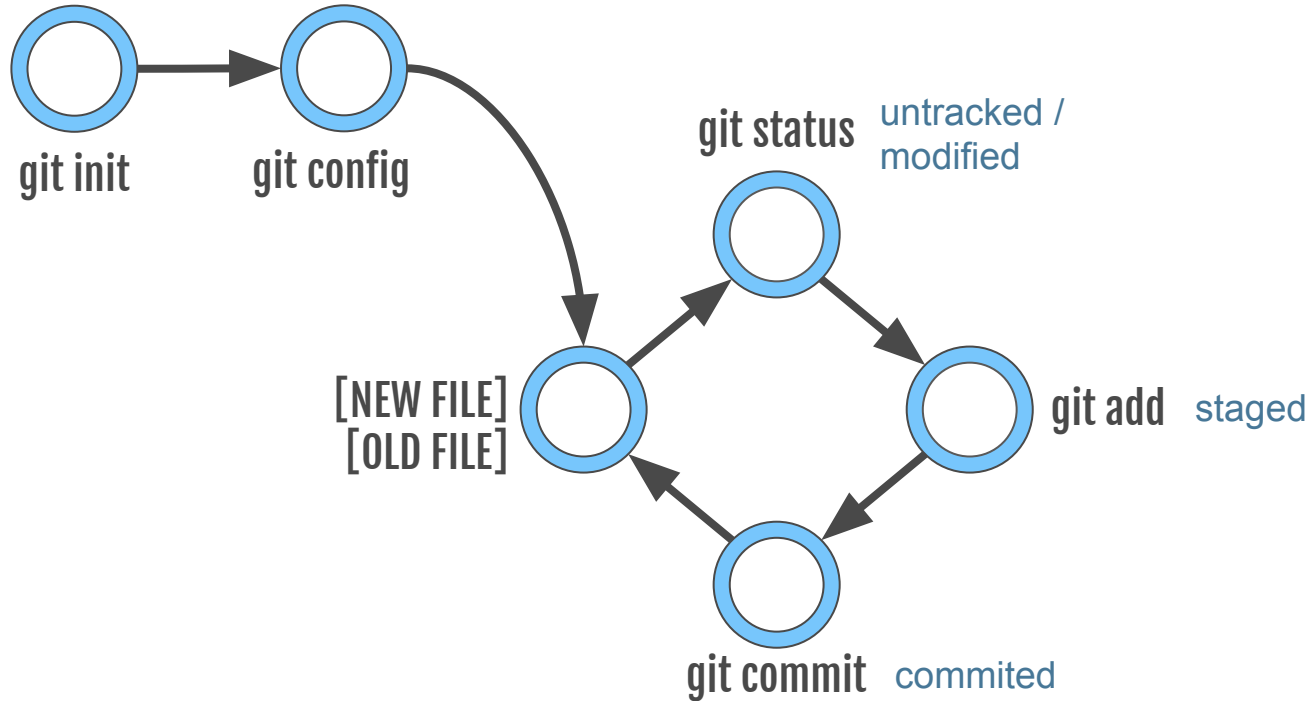
A diagram illustrating a Git workflow. A large, light blue, irregular blob is centered on the page, containing the text "git workflow" in a bold, dark gray, lowercase sans-serif font. Surrounding this central blob are several circular nodes connected by thin black lines, representing a commit history. Each node consists of two concentric circles: an outer light blue ring and an inner white circle with a small blue dot in the center. The nodes are arranged in a branching pattern. In the top right, a sequence of three nodes is connected by lines, with an ellipsis (...) above the final node. In the bottom left, a sequence of four nodes is connected by lines, with an ellipsis (...) above the first node. The overall background is a very light gray.

git workflow

01 - Introdução ao controle de versão



01 - Introdução ao controle de versão



01 – Introdução ao controle de versão

Boas práticas para um commit

- Primeira linha com não mais de 50 caracteres
- Pule a segunda linha
- Da terceira para frente, descreva com o máximo de informações técnicas possíveis o que foi realizado na modificação. Lembre-se o próximo a pegar o código, com certeza, será uma pessoa diferente da que pegou agora.
- Siga padrões de commit utilizados pela comunidade.

<https://www.conventionalcommits.org/pt-br/v1.0.0/>

Commit c97fa9b2  authored há um mês por

fix

Provide a good commit message example

The purpose of this commit is to provide an example of a hand-crafted, artisanal commit message. The first line is a short, approximately 50 character summary, followed by an empty line. The subsequent paragraphs are jam-packed with descriptive information about the change, but each line is kept under 72 characters in length.

If even more information is needed to explain the change, more paragraphs can be added after blank lines, with links to issues, tickets, or bugs. Remember that future you will thank current you for your thoughtfulness and foresight!

```
# Please enter the commit message for your changes. Lines starting
# with '#' will be ignored, and an empty message aborts the commit.
#
# On branch master
#
# Changes to be committed:
# new file:   super_script.py
# new file:   cool_config.txt
#
```



A diagram illustrating a Git log. A large, light blue, irregular blob shape is centered on the page. Inside this blob, the text "git log" is written in a bold, dark gray, sans-serif font. Surrounding the central blob are several circular nodes connected by thin black lines, representing a commit history. Each node is a circle with a blue outline and a smaller blue circle inside. Some nodes are larger than others. In the top right, a sequence of three nodes is shown, with the third being the largest and having three dots below it. In the bottom left, a sequence of four nodes is shown, with the second being the largest and having three dots above it. The overall style is clean and modern, using a light gray background.

git log



Thanks!

Até a próxima!
13 de novembro
14h

